NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TRAJECTORY OPTIMIZATION FOR HELICOPTER UNMANNED AERIAL VEHICLES (UAVs)

by

Benjamin Thomas Gatzke

June 2010

Thesis Advisor:	Wei Kang
Second Reader:	Hong Zhou

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.		
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Trajectory Optimization for Helicopter Unmanned Aerial Vehicles (UAVs)		**5. FUNDING NUMBERS**
6. AUTHOR(S) Benjamin Thomas Gatzke		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis explores the numerical methods and software development for optimal trajectories of a specific model of Helicopter Unmanned Aerial Vehicle (UAV) in an obstacle-rich environment. This particular model is adopted from the UAV Laboratory of the National University of Singapore who built and simulated flights for an X-Cell 60 small–scale UAV Helicopter. The code, which allowed the team to simulate flights, is a complex system of non-linear differential equations—15 state variables and four control variables—used to maneuver the state trajectories. This non-linear model is incorporated into a separate optimization algorithm code, which allows the user to set initial and final time conditions together with various constraints, and, using the same variable scheme, optimize a trajectory. The optimal trajectory is defined by using a cost function—the performance measure—and the system is subject to a set of constraints (such as mechanical limitations and physical three-dimensional obstacles). Simulations conclude that solutions are readily obtained; however, it is still very difficult to derive trajectories that are truly optimal, and our work calls for more future research in computational programs for optimal trajectory planning. All simulations in this thesis are modeled using the MATLAB program.

14. SUBJECT TERMS Nonlinear Model, Trajectory Optimization, State and Control Variables, Cost Function			**15. NUMBER OF PAGES** 77
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

TRAJECTORY OPTIMIZATION FOR HELICOPTER UNMANNED AERIAL VEHICLES (UAVs)

Benjamin T. Gatzke
Major, United States Army
B.S., United States Military Academy at West Point, May 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2010

Author:	Benjamin Thomas Gatzke
Approved by:	Wei Kang Thesis Advisor
	Hong Zhou Second Reader
	Carlos Borges Chairman, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the numerical methods and software development for optimal trajectories of a specific model of Helicopter Unmanned Aerial Vehicle (UAV) in an obstacle-rich environment. This particular model is adopted from the UAV Laboratory of the National University of Singapore who built and simulated flights for an X-Cell 60 small-scale UAV Helicopter. The code, which allowed the team to simulate flights, is a complex system of non-linear differential equations—15 state variables and four control variables—used to maneuver the state trajectories. This non-linear model is incorporated into a separate optimization algorithm code, which allows the user to set initial and final time conditions together with various constraints, and, using the same variable scheme, optimize a trajectory. The optimal trajectory is defined by using a cost function—the performance measure—and the system is subject to a set of constraints (such as mechanical limitations and physical three-dimensional obstacles). Simulations conclude that solutions are readily obtained; however, it is still very difficult to derive trajectories that are truly optimal, and our work calls for more future research in computational programs for optimal trajectory planning. All simulations in this thesis are modeled using the MATLAB program.

.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The success of this thesis and my Master's degree completion at the Naval Postgraduate School are due, in great part, to the generosity, hospitality, and tremendous attitude of the entire Department of Applied Mathematics, whose professionalism is unparalleled. Particularly, I owe a great deal of gratitude to my thesis advisor, Dr. Wei Kang, whose expertise in the field of optimal control, coupled with an awesome level of patience, have enabled me to accomplish this project. I must also thank MAJ Joe Lindquist for his selfless nature and constant willingness to help my never-ending string of coding issues. I also appreciate the help from CDR (Ret.) Les Carr for helping me understand how a helicopter works. Thanks are also in order for LTC (Ret.) Bard Mansager for allowing me the opportunity to study here and to Dr. Carlos Borges for the bi-weekly "counseling" sessions and the endless positive reinforcement that all students should receive from their department's leadership. Finally, as always, I must thank my family for their constant love and support through everything in my 11 years in the Army.

THIS PAGE INTENTIONALLY LEFT BLANK

I. LITERATURE REVIEW

A. BACKGROUND

This project explores and simulates a specific algorithm for trajectory optimization for UAVs in an environment with fixed or unfixed obstacles that may be hard or soft in nature. A hard obstacle represents a physical obstruction where contact will result in damage to the system and mission failure. A soft obstacle represents something that should be avoided, but the algorithm user can designate a priority or weighting to the obstacle to portray its level of detriment to the UAV. Since both path and trajectory planning are popular topics of research among the science community—with a plethora of applications—it is important to conduct a literature review, both to gather ideas that may be helpful to research and simulation, and to ensure that our algorithm is unique in nature.

There are two types of relevant problems in the reviewed literature: path planning and trajectory planning and optimization. Path planning is the least complex of the two, but may provide some simple ideas that can still be applied to more complex problems. Trajectory planning and trajectory optimization are very similar as far as variables and parameters are concerned. The main difference is that trajectory optimization is focused on maximizing or minimizing a specific cost function (i.e., travel time, cost, fuel consumption). This chapter provides a brief description of pertinent articles researched in each of the two categories.

B. ARTICLE REVIEWS

Path planning is the simplest form of planning, since it generally involves directional components of a vehicle or aircraft, which means that the model will consist of two space variables for 2-D and three space variables for 3-D. Occasionally, an analysis of path planning will include velocities which would increase the number of variables to four and six, respectively. The best path is created by optimizing these variables over a set of constraints.

In [1], a team analyzes a problem slightly more difficult than standard path planning, as it describes a method for finding the shortest path for multiple UAVs flying simultaneously, rather than just one aircraft. This team uses Dubins paths, which calculate a sum of circular arcs and their tangent lines, to determine the best paths for a swarm of UAVs to simultaneously converge on a target region. The model uses three general constraints: (1) A maximum bound on the UAV curvature (due to limitations of the aircraft), (2) Minimum separation distance between UAVs, and (3) Non-intersection of paths at equal length (to prevent collision). The model intends to protect the aircraft while minimizing distance travelled, which inherently achieves the goals of minimizing fuel consumption while increasing durability of the UAVs [1].

Each UAV path is generated through finding a common plane between the initial and final position vectors and the final tangent vector (i.e., for each path, the three vectors are co-planar). Then, each path length is calculated using the a specific set of equations. Each path is compared to a “Reference Path” which is the longest of all possible UAV paths. Each path is then lengthened (by reducing curvature) and compared against all existing constraints until an optimal solution is obtained. Since no aircraft specifications pertain to this simple model, it may be used for fixed or rotary winged aircraft [1].

The aspects of this article are helpful, since they provide a method for coordinating several UAVs for simultaneous arrival on a designated target; however, they avoid several critical aspects that are better addressed in trajectory planning and optimization. First, this model assumes an equal path length and a constant speed for all UAVs. Second, the analysis assumes an obstacle-free environment. Finally, the equations used for this path planning method do not optimize any critical performance functions that, in turn, do not allow any maximization/minimization planning effects [1].

Models of trajectory planning and/or optimization become more complicated since they incorporate dynamics involving measures of performance of the actual vehicle or aircraft, as well as directional variables. While parameters and constraints still exist in the model, the number of variables and dimension of the differential equation system will increase substantially. In this case, the nonlinearity in the dynamics becomes the main challenge in the searching for optimal solutions.

In [2], the authors study a time-variant model by analyzing the probability of a UAV becoming disabled at a certain time juncture in a dynamic environment. In other words, the probability of a UAV becoming disabled varies over the time variable. This particular model requires that the UAV reach the objective (accomplishing the mission) while utilizing the shortest path possible or finding the trajectory that minimizes the probability of the UAV being disabled. Weighting of parameters by the user determines which criterion is most important to the model. The probabilistic map changes constantly over time; therefore, the paths generated by this strategy incorporates functions of both position and time. Consequently, variables will exist for velocity and acceleration, not just for the position vector (as is the case in path planning). That means that this model must also incorporate several constraints for UAV capabilities. As a UAV flies in an area with multiple threats, the risk of the UAV becoming disabled is characterized by the probability of the UAV becoming disabled at a certain location and time. The probability is modeled using Gaussian Probability Density Functions [2].

This probabilistic approach provides a much more useful method for UAV planning, using trajectory planning rather than path planning. This allows the modeler to emplace constraints based on the capabilities of the UAV and capabilities of the threat as well as path direction. The model also allows all variables to change over time which means that obstacles can be fixed, moving, and/or with changing capabilities based on both time and position. These aspects of this model are very useful for the prospect of real-time trajectory planning [2].

In [3], CDR Mike Hurni analyzes optimal control techniques for vehicle trajectories specifically pertaining to minimizing the cost function for ground vehicle operations. His dissertation shows, very methodically, how optimal control techniques can work effectively to minimize cost (or another user selected performance measure) while maximizing the flexibility for a ground vehicle to alter its path in a real-time fashion in response to obstacles known or not known *a priori*. In other words, his techniques can minimize cost while maximizing robustness. His techniques also provide a great deal of flexibility in allowing the user to alter weighting values to increase, decrease, or eliminate the impact of certain aspects of the trajectory on the overall cost function value [3].

Additionally, CDR Hurni's work provides a series of "requirements checks" on a particularly selected trajectory to test feasibility, obstacle collision/buffer violation, caution zone infractions, and several other constraints that may deem the trajectory to cause mission failure. These checks are established in a way to allow the user to vary buffer zones, the number of obstacles, the state of obstacles—fixed or unfixed—and other key criteria that are critical to the flexibility of this particular optimal control system [3].

In the final stages of the dissertation, CDR Hurni addresses the prospect of multiple ground vehicles operating in the same space simultaneously. His algorithm allows the user to implement weights involving collision and grazing prevention for a varying number of vehicles. These weights are in addition to the previously calculated weights for obstacle avoidance. This addition allows an even broader range of use for the algorithm. CDR Hurni's use of optimal control methods has resulted in an extremely flexible algorithm that allows a user to input and alter numbers and weightings for obstacle and ground vehicle criteria and constraints while optimizing trajectory to minimize or maximize a function selected by the user. The only disadvantage to this project, as it pertains to our work, is that our problem deals with a nonlinear system and much more complex dynamics [3].

In [4], a team documents their construction of a vehicle to compete in the DARPA Urban Challenge by formulating and solving an optimal and real-time trajectory planning problem with velocity variables and nonlinear dynamics. In forming their optimal trajectory formula, the team integrates boundary conditions at the initial and final time instances, motion constraints, and collision avoidance criteria as the three conditions that must be satisfied to enable a feasible trajectory. The fourth constraint is the minimization of the "performance index" (optimization). These constraints are designed with the assumption that boundary conditions and motion constraints are generally given while optimization and collision avoidance criteria are chosen to fit the designer's needs for a specific situation [4].

The key development for this trajectory planning problem is the idea of real-time obstacle avoidance. This means that it is assumed that obstacles may be fixed or moving and that the positions of these obstacles are generally not known a priori which means that trajectories must be re-planned multiple times throughout movement between the initial and final positions. Three key features enable this method to satisfy the requirement: (1) All paths satisfying boundary conditions and the vehicle's kinematic constraints are parameterized in terms of polynomials of sufficient order, (2) A collision-free criterion is developed and imposed for avoiding both "hard" and "soft" obstacles that are detected along the path (in real time), and (3) A performance index is introduced to find the best path among the collision-free paths meeting all necessary criteria [4]. Finally, the performance index is chosen so that paths equivalent to the shortest path can be solved analytically while meeting all criteria—based on both given constraints and those imposed by the designer. The bottom line is that, for a successfully optimal trajectory that completes the mission in a real-time changing dynamic environment, the trajectory must be modeled in a piecewise fashion as the vehicle moves from its initial to final position. The team's conclusion is that this is most effectively accomplished through use of a parameterized fourth-order polynomial. By obtaining the solution to an adjustable parameter, it is possible to generate a real-time solution [4].

The method discussed in this article provides the same advantages as were discussed in CDR Hurni's dissertation with the exception of not including a model with multiple vehicles to be coordinately controlled. Also, the solution for this model was derived analytically, and our complex nonlinear model will most certainly require a numerical model to ensure solvability.

C. CONCLUSION

The professional works uncovered during the literature review phase of this thesis have shown that trajectory optimization is a complex problem which complicates drastically when non-linear dynamics and constraints are involved. The papers reviewed demonstrate several methods for analyzing and solving path and trajectory planning problems. They all have strengths and weaknesses that are pertinent to our future research. The key point is that none of them truly solve our exact situation that requires optimality of the performance measure, so we will surely incorporate some of the ideas from these works. Significant original input will also be required to fulfill our objective: to derive and test an algorithm to numerically solve an optimal trajectory for a UAV in a dynamic environment with and without obstacles.

II. PROBLEM FORMULATION

A. INTRODUCTION

At the end of Chapter I, it is clearly stated that the objective is to produce an optimal UAV trajectory in a dynamic environment with fixed and/or moving obstacles. This objective requires the derivation of a system of equations and functions that consider three separate elements: 1) The dynamic equations that determine the trajectory of the UAV, 2) The trajectory constraints, and 3) The performance measure for optimization. The dynamic equations involve a system of both state and control vectors. The state vectors are represented by the positions, attitudes, and velocities of the UAV—all with respect to time. The control vectors are determined by the system inputs controlled by the algorithm's user. Trajectory constraints, in this case, alter the trajectory to avoid all obstacles in a projected path. The optimizing performance measure outlines a function to minimize cost, time, or another parameter of our choosing by solving for the optimal control vector in a given trajectory. The remainder of this section will use the three elements explained above to develop a logical and solvable problem that allows us to fulfill the stated objective.

B. TRAJECTORY EQUATIONS

The trajectory equations for this particular system will be structured in the form of a comprehensive non-linear differential equation model based on a Helicopter UAV built and modeled by a research team from the National University of Singapore. The model will consist of four key components that will derive a total of 15 state variables and, consequently, 15 differential equations: 1) Kinematics, 2) Six degree-of-freedom (DOF) rigid-body dynamics, 3) Main rotor flapping dynamics, and 4) Yaw rate gyro dynamics [5]. Prior to introducing the non-linear system, Table 1 defines the 15 state variables and four control variables. First, it is important to realize that the UAV will operate in two different sets of coordinate frames: Body Frame (used for velocities and based on coordinates with reference to the actual vehicle) and North-East-Down Frame (used for positions and based on an initially fixed set of coordinates [North = x, East = y, Down = z]) [5].

Table 1. Dynamic Equation Variable Definitions

Variable	Variable Definition	Unit
p_x, p_y, p_z	Position Vector along the North-East-Down (NED) Frame (x,y,z)	meters
u, v, w	Velocity Vector along the Body Frame (x,y,z)	meters/sec.
Φ, θ, Ψ	Roll, Pitch, and Yaw Angles (Euler Angle in the NED Frame)	radians
q, r, s	Roll, Pitch, and Yaw Angular Rates in the Body Frame	radians/sec.
a_s, b_s	Longitudinal and Lateral Tip-Path-Plane (TPP) Flapping Angles	radians
$\delta_{ped,\text{int}}$	Intermediate State in Yaw Rate Gyro Dynamics	N/A
δ_{lat}	Allows the user to control the ailerons of the Helicopter UAV	N/A
δ_{lon}	Allows the user to control the elevators of the Helicopter UAV	N/A
δ_{col}	Allows the user to control the collective pitch of the Helicopter UAV	N/A
δ_{ped}	Allows the user to control the rudder of the Helicopter UAV	N/A

1. Kinematics

The kinematics component generates six of the 15 state variable differential equations for our system. The six equations are derived from two vector equation systems, each with three equations, based on position and velocity (in the x, y, and z directions). The first equation set is for position:

$$\dot{P}_n = B_B \cdot V_b$$

where

$$P_n = (p_x, p_y, p_z)^T$$

is the position vector in the NED Frame and

$$V_b = (u, v, w)^T$$

is the velocity vector in the Body Frame. Since the position and velocity vectors are defined in different coordinate spaces, it is imperative to have a transformation matrix between the two. This transformation matrix, designated B_B, is defined below [5].

$$B_B = \begin{pmatrix} \cos(\theta)\cos(\Psi) & \sin(\Phi)\sin(\theta)\cos(\Psi) - \cos(\Phi)\sin(\Psi) & \cos(\Phi)\sin(\theta)\cos(\Psi) + \sin(\Phi)\sin(\Psi) \\ \cos(\theta)\sin(\Psi) & \sin(\Phi)\sin(\theta)\cos(\Psi) + \cos(\Phi)\cos(\Psi) & \cos(\Phi)\sin(\theta)\sin(\Psi) - \sin(\Phi)\cos(\Psi) \\ -\sin(\theta) & \sin(\Phi)\cos(\theta) & \cos(\Phi)\cos(\theta) \end{pmatrix}$$

The second kinematic equation set is for rotational motion:

$$\dot{\Omega}_n = S_B \cdot \Omega_b$$

where

$$\Omega_n = (\Phi, \theta, \Psi)^T$$

is the Euler Angle Vector in the NED Frame and

$$\Omega_b = (q, r, s)^T$$

is the angular velocity vector in the Body Frame. Just as is the case with the first equation set, the different coordinate spaces require a corresponding transformation matrix, designated S_B and defined below.

$$S_B = \begin{pmatrix} 1 & \tan(\theta)\sin(\Phi) & \tan(\theta)\cos(\Phi) \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \dfrac{\sin(\Phi)}{\cos(\theta)} & \dfrac{\cos(\Phi)}{\cos(\theta)} \end{pmatrix}$$

2. Rigid-Body Dynamics

The six degrees of freedom of the rigid body dynamics of the helicopter generate two additional systems of equations, with each system containing three equations. These six equations derive the six components of the helicopter's velocity (three in space and three angular) and are represented by the following Newton-Euler Equations:

$$\dot{V}_b = (-\Omega_b \times V_b) + \frac{F_b}{m} + \frac{F_g}{m}$$

and

$$\dot{\Omega}_b = I^{-1}[M_b - (\Omega_b \times I \bullet \Omega_b)]$$

where

$$F_g = \left(mg \sin\theta, mg \sin\Phi \cos\theta, mg \cos\Phi \cos\theta\right)^T$$

is the gravity force vector,

$$F_b = \begin{bmatrix} F_{bx} \\ F_{by} \\ F_{bz} \end{bmatrix} = \begin{bmatrix} (X_{mr} + X_{fus}) \\ (Y_{mr} + Y_{fus} + Y_{tr} + Y_{vf}) \\ (Z_{mr} + Z_{fus} + Z_{hf}) \end{bmatrix}$$

is the aerodynamic force vector,

$$M_b = \begin{bmatrix} M_{bx} \\ M_{by} \\ M_{bz} \end{bmatrix} = \begin{bmatrix} (L_{mr} + L_{vf} + L_{tr}) \\ (M_{mr} + M_{hf}) \\ (N_{mr} + N_{vf} + N_{tr}) \end{bmatrix}$$

is the moment vector, and

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

is the moment of inertia matrix. The term *m* represents the total mass of the helicopter UAV, and g represents a constant value for gravity. The subscripts *mr, tr, fus, vf,* and *hf* represent main rotor, tail rotor, fuselage, vertical fin, and horizontal fin or the helicopter, respectively. The *X, Y, Z, L, M, and N* terms are all equations based on combinations of different coefficients and variables. Expanded detail of all terms can be found in [5].

3. Main Rotor Flapping Dynamics

These two state variables, a_s and b_s, represent the longitudinal and lateral Tip Path Plane angles as the main rotor spins during flight. The dynamics of these two variables are represented by the following equations:

$$\dot{a}_s = -r - \frac{a_s}{\tau} + \frac{1}{\tau}\left(\frac{\delta a_s}{\delta u}u + \frac{\delta a_s}{\delta w}w\right) + \frac{A_{\delta_{lon}}}{\tau}\delta_{lon}$$

and

$$\dot{b}_s = -q - \frac{b_s}{\tau} - \frac{1}{\tau}\frac{\delta b_s}{\delta v}v + \frac{B_{\delta_{lat}}}{\tau}\delta_{lat}$$

where τ represents the effective rotor time constant of the main rotor system, $\frac{\delta a_s}{\delta u}$ and $\frac{\delta b_s}{\delta v}$ are the longitudinal and lateral dihedral effect derivatives, $\frac{\delta a_s}{\delta w}$ is the flap-back effect derivative, $A_{\delta_{lon}}$ is the effective linkage gain from the control variable δ_{lon} to the cyclic pitch angle along the longitudinal direction, and $B_{\delta_{lat}}$ is the effective linkage gain from the control variable δ_{lat} to the cyclic pitch angle along the longitudinal direction [5]. Control variables are defined and explained in Section C. Expanded detail of these terms can be found in [5].

4. Yaw Rate Gyro Dynamics

For small-scale remote controlled helicopters, the yaw moment is very sensitive; therefore, it is very difficult to control yaw motion in manual flight. To combat this challenge, small-scale helicopters are commonly equipped with a yaw rate gyro, which consists of a gyro sensor and an embedded controller, to allow the human controller to modify the yaw rate and/or heading. In modern models, such a device is not essential to fly a programmed path; however, it is reserved for the manual back-up flight system. Therefore, the yaw rate gyro's dynamics must be included in the non-linear flight model and are represented by the following differential equation:

$$\dot{\delta}_{ped,\mathrm{int}} = K_a \delta_{ped} - s$$

where K_a represents a scaling value (a constant of value 3.73), and δ_{ped} represents the control variable for rudder servo input (defined and explained in Section C) [5].

The 15 variables described in the four preceding subsections are the system whose solutions, based on the variation of the four control variables, were used by the team in the non-linear model to code simulated flight paths for the X-Cell 60 small-scale UAV helicopter. This same non-linear model is the critical input piece into the code that will eventually create a numerical program to optimize the trajectory of the same UAV helicopter.

C. CONTROLS

In order to achieve solutions to the 15 state variable differential equations, it is essential to impose a set of controls that will directly impact the flight of the helicopter and, therefore, will generate state variable solutions throughout an interval of time. These controls, which are non-dimensional, have been scaled in this particular model to have values between -1 and 1. There are four total control variables, and they are annotated, along with their impact on the helicopter's flight, in the following subsections [5].

1. Aileron Servo Input (δ_{lat})

This control variable allows the user to control the ailerons of the helicopter. This particular input will directly influence the state of roll for the aircraft (*p* and Φ as far as the state variables are concerned) [5].

2. Elevator Servo Input (δ_{lon})

This control variable allows the user to control the elevators of the helicopter. This particular input will directly influence the state of pitch for the aircraft (*q* and θ as far as the state variables are concerned) [5].

3. Rudder Servo Input (δ_{ped})

This control variable allows the user to control the rudder of the helicopter. This particular input will directly influence the state of yaw for the aircraft (*r* and Ψ as far as the state variables are concerned) [5].

4. Collective Pitch Servo Input (δ_{col})

The collective pitch control is the most unique of the four. While it has the same range of input values, it is not an independent control. This means that it may not be altered without changing at least one of the other control variables to balance it. Changing only the collective pitch will cause the aircraft to become unstable and possibly crash. Obviously, this provides disastrous results for trajectory simulations [5].

The user programmed inputs of the aforementioned four control variables, forced to conform to pre-determined constraints, will allow a property designed non-linear model to create solutions to the state variable differential equation system.

D. CONSTRAINT EQUATIONS

The development of obstacle constraints involves limitations to the trajectory of the vehicle that do not involve maximum and minimum restrictions from the system state or control inputs. This section will focus on obstacle avoidance.

The main focus of this particular UAV Trajectory Model is for use in an urban environment, so it is important to consider typical obstacles one might encounter while travelling aerially in a city. There are three obstacle types: buildings, power lines (with poles), and bridges. Ideally, a model that can handle all three types is optimal. It is important to note that the model cannot produce obstacles with rigid corners, as this can result in non-differentiable functions as we attempt to solve the system. The obstacle modeling in this section uses the p-norm to create a variation on the standard 3-D equation for an ellipsoid-shaped region. The equation *h(x,y,z)* shows the general form used to model the obstacles.

$$h(x(t),y(t),z(t)) = \left|\left(\frac{x(t)-x_\varphi}{a}\right)^p\right| + \left|\left(\frac{y(t)-y_\varphi}{b}\right)^p\right| + \left|\left(\frac{z(t)-z_\varphi}{c}\right)^p\right| - 1 = 0$$

- *x(t), y(t), z(t)* represent the position (x,y,z) with respect to time
- $x_\varphi, y_\varphi, z_\varphi$ represent the center position locations inside of each obstacle
- *a* represents the distance from the center to the obstacle boundary in the x-direction
- *b* represents the distance from the center to the obstacle boundary in the y-direction

- *c* represents the distance from the center to the obstacle boundary in the z-direction [3].

The absolute value signs can be removed from the equation if the p-values are limited to even numbers. This is suitable, since increasing the value of *p* will merely alter the shape of the obstacle from more circular/elliptical to more square/rectangular [3]. Therefore, the equation below, where *p>0* and even, will be used for all subsequent examples which will model several obstacles types using MATLAB.

$$h(x(t),y(t),z(t)) = \left(\frac{x(t)-x_\varphi}{a}\right)^p + \left(\frac{y(t)-y_\varphi}{b}\right)^p + \left(\frac{z(t)-z_\varphi}{c}\right)^p - 1 = 0$$

This model assumes that the ground is flat and represents the *xy*-plane which means that the *z*-axis is vertical and $z \geq 0$ in all instances.

Example 1

$$\left(\frac{x}{4}\right)^{20}+\left(\frac{y}{4}\right)^{20}+\left(\frac{z-2}{2}\right)^{20}-1=0$$

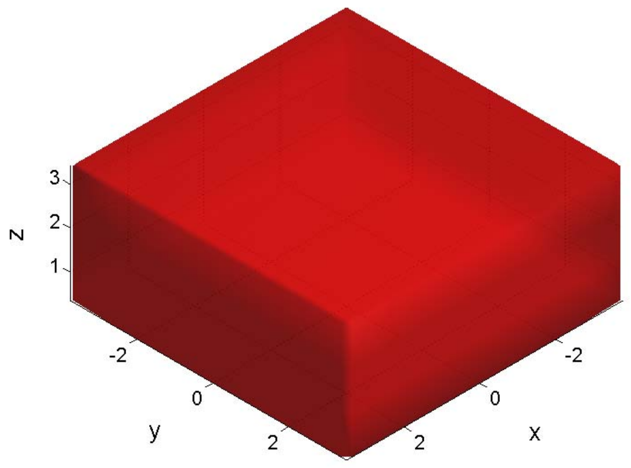

Figure 1. A short, block building with center point (0,0,2) and a=4, b=4, c=2, p=20

Example 2

$$\left(\frac{x}{4}\right)^{40}+\left(\frac{y}{6}\right)^{40}+\left(\frac{z-15}{15}\right)^{40}-1=0$$

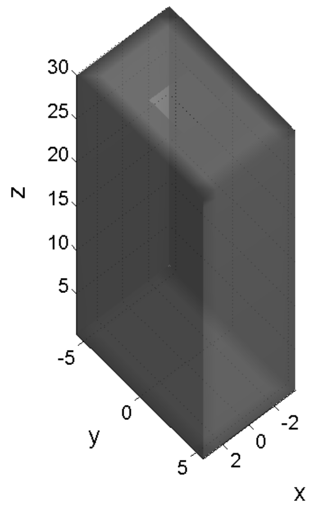

Figure 2. A skyscraper type of building with center point (0,0,15) and a=4, b=6, c=15, p=40

Example 3

Left Pillar: $$\left(\frac{x-29}{2}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z-10}{20}\right)^{20}-1=0$$

Right Pillar: $$\left(\frac{x+29}{2}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z-10}{20}\right)^{20}-1=0$$

Bridge Portion: $$\left(\frac{x}{30}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z-5}{2}\right)^{20}-1=0$$

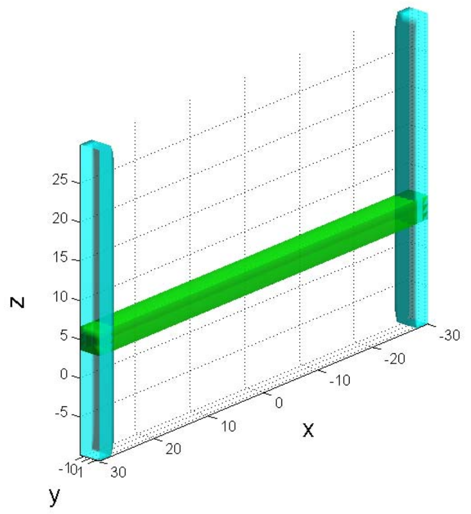

Figure 3. A flat bridge with two end pillars (plot is constructed using three separate functions; one for each pillar and one for the bridge portion)

Example 4

Left Pillar: $$\left(\frac{x-29}{2}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z+35}{5}\right)^{20}-1=0$$

Right Pillar: $$\left(\frac{x+29}{2}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z+35}{5}\right)^{20}-1=0$$

Bridge Portion: $$\left(\frac{x}{30}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z+30}{2}\right)^{20}-1=0$$

Suspension Supports:

$$\left(\frac{x+15}{15}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z+|2(x+15)|}{20}\right)^{20}-1=0 \quad \text{for } x<0$$

$$\left(\frac{x+15}{15}\right)^{20}+\left(\frac{y}{2}\right)^{20}+\left(\frac{z+|2(x+15)|}{20}\right)^{20}-1=0 \quad \text{for } x>0$$

Suspension Cables:

$$\left(\frac{x\pm d}{.5}\right)^{2}+\left(\frac{y}{2}\right)^{2}+\left(\frac{z\pm(30-c)}{c}\right)^{20}-1=0$$

Note: For the Suspension Cable Equation, a value of c must be selected so that the z coordinate for the to of the suspension cable matches the z coordinate for the suspension support for each selected value of x±d.

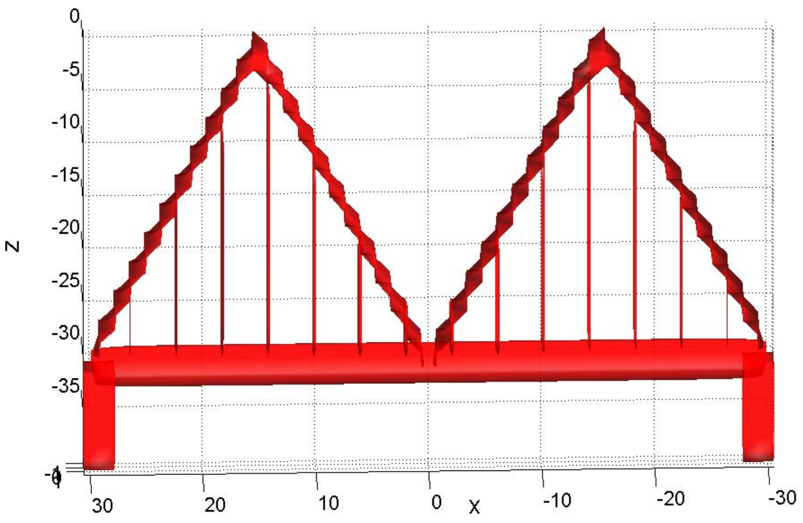

Figure 4. A suspension bridge with two end pillars below the road level, two triangular suspension supports above the road level, and a number of cables (number and positions along the x-axis can be altered easily by the designer within the equation)

Examples 1 through 4 demonstrate legitimate urban obstacle possibilities with constants that the designer can easily alter to change the size, shape, and position of the particular obstacles. Although the implicit equation used is exponential in nature, most of the outcomes are fairly linear in nature; another challenge arises when we concern obstacles with curvature—i.e., power lines. This requires the orientation of the previous model in the direction of a Catenary. To form the suspension supports in Example 4, a variant of the third term, using a linear x-term in addition to the z-term, provides the necessary result. Example 5 demonstrates an example of a power line obstacle, but, contrary to Example 4, the x-term added to the third term is non-linear in nature, which inherently results in curvature.

Example 5

Left Pole: $$\left(\frac{x-29}{2}\right)^2+\left(\frac{y}{2}\right)^2+\left(\frac{z+2}{15}\right)^{20}-1=0$$

Right Pole: $$\left(\frac{x+29}{2}\right)^2+\left(\frac{y}{2}\right)^2+\left(\frac{z+2}{15}\right)^{20}-1=0$$

Power Lines: $$\left(\frac{x}{30}\right)^{10}+\left(y\right)^{10}+\left(z-1-\frac{\left|x^c\right|}{20c}\right)^2-1=0 \quad \text{for } c\in[0.5,2]$$

Note: The larger the value of c, the more increased the curvature in the shape becomes. At values of c>2, the plot in MATLAB begins to disintegrate. Values of c>2 should be unnecessary for the purpose of designing obstacle for this project. For the plot below, c = 1.6.

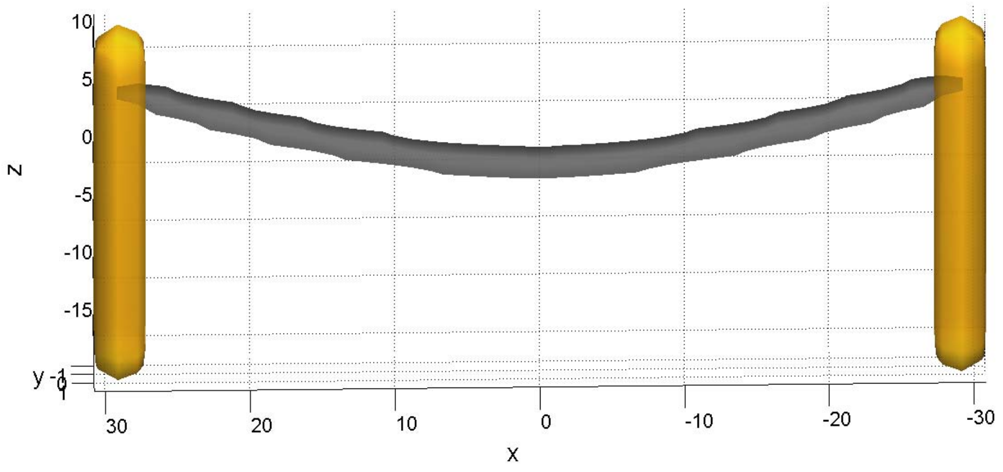

Figure 5. A set of power lines with two poles and a buffer area simulating the curved area created by hanging lines

Examples 1 through 5 have demonstrated some different equations for modeling certain types of common urban obstacles into the constraint function.

E. THE PERFORMANCE MEASURE

Thus far, we have discussed the development of a non-linear state variable equation system, using control variables and constraints, for a Helicopter UAV model that allows a user to simulate flights and plot the trajectories. A user may alter these trajectories through changing the control variables in the computer model. Finally, an optimization function will incorporate the non-linear simulation model, and all of its conditions, into an integral (evaluated over time) that will allow the user to minimize a particular aspect of the UAV's operation—called a cost.

In order to complete the problem formulation process for optimal control, it is critical to identify clearly the performance measures of the system one wishes to optimize. A different designer may choose different performance measures, but for this project, we will use *time* and *cost*. An optimization function is one that will either minimize or maximize the chosen performance measures—in this case, both time and cost will be *minimized*. The optimization functions developed in this section will combine the states, controls, and constraints into one equation that eventually provides our desired optimization endstate.

The problem in this model finds a solution for the optimal control vector (**u***) which causes the system

$$\frac{d\vec{x}(t)}{dt} = f(\vec{x}(t), \vec{u}(t), t)$$

to follow an allowable trajectory (**x***) which minimizes the performance measure (J) such that

$$J = h(\vec{x}(t_f), t_f) + \int_{t_0}^{t_f} f(\vec{x}(t), \vec{u}(t), t)dt$$

where the function $h(\vec{x}(t_f), t_f)$ represents an endpoint cost not associated with the trajectory itself [6].

In this case, the allowable trajectory **x*** that minimizes the performance measure J is called the optimal trajectory [6]. Two things must be realized prior to modeling optimal control problems. First, an optimal control solution may not be unique. This is not necessarily detrimental; although it may complicate the computational aspect of calculation, it can allow flexibility for the designer's configuration scheme. Second, an optimal control may not exist—this is detrimental to our desired results

In order for the constraint algorithm to perform properly, four critical input components are coded to formulate a solution:

- The Cost Function: Defines and codes the integral for the actual cost optimization function
- The Dynamics Function: Takes the inputted non-linear model from the flight simulation codes to program the state and control differential equation systems into the optimization algorithm
- The Path Function: Allows the user to program the desired constraints into the optimization algorithm in order to limit state variable outputs
- The Events Function: Defines and codes the initial and final condition limits for state variable values on which the user desires to enforce hard values.

The next part of this project will describe how the algorithm inside of the optimization programs works to produce a solution from the aforementioned inputs.

Overall, this thesis will attempt to obtain a unique optimal solution for, at a minimum, several examples of the following scenario for the X-Cell 60 small-scale helicopter UAV:

- Minimize flight time; no obstacles.

In summary, we will find an optimal solution (minimum flight time) for *J* in

$$J = \int_{0}^{t_f} f(\vec{x}(t), \vec{u}(t), t)\,dt$$

where

$$\vec{x}(t) = \begin{pmatrix} p_x \\ p_y \\ p_z \\ u \\ v \\ w \\ \Phi \\ \theta \\ \Psi \\ q \\ r \\ s \\ a_s \\ b_s \\ \delta_{ped,\mathrm{int}} \end{pmatrix}$$

and

$$\vec{u}(t) = \begin{pmatrix} \delta_{lat} \\ \delta_{lon} \\ \delta_{col} \\ \delta_{ped} \end{pmatrix}$$

governed by

$$\dot{P}_n = B_B \cdot V_b$$

$$\dot{\Omega}_n = S_B \cdot \Omega_b$$

$$\dot{V}_b = (-\Omega_b \times V_b) + \frac{F_b}{m} + \frac{F_g}{m}$$

$$\dot{\Omega}_b = I^{-1}[M_b - (\Omega_b \times I \bullet \Omega_b)]$$

$$\dot{a}_s = -r - \frac{a_s}{\tau} + \frac{1}{\tau}\left(\frac{\delta a_s}{\delta u} u + \frac{\delta a_s}{\delta w} w \right) + \frac{A_{\delta_{lon}}}{\tau} \delta_{lon}$$

$$\dot{b}_s = -q - \frac{b_s}{\tau} - \frac{1}{\tau} \frac{\delta b_s}{\delta v} v + \frac{B_{\delta_{lat}}}{\tau} \delta_{lat}$$

and

$$\dot{\delta}_{ped,\text{int}} = K_a \delta_{ped} - s$$

where

$$P_n = (p_x, p_y, p_z)^T$$

$$V_b = (u, v, w)^T$$

$$\Omega_n = (\Phi, \theta, \Psi)^T$$

$$\Omega_b = (q, r, s)^T .$$

THIS PAGE INTENTIONALLY LEFT BLANK

III. PSEUDOSPECTRAL METHOD FOR OPTIMAL CONTROL

A. INTRODUCTION

For quite some time, mathematicians have struggled with a reliable method for solving optimal control problems with complicated nonlinear dynamics and constraints. Over the past decade, direct computational methods have become increasingly popular for solving these optimal control problems [7]. In direct methods, a continuous time problem is discretized over a set time interval, and the resulting discrete problem is solved numerically using nonlinear programming algorithms. Due to significant progress in large-scale computational algorithms and nonlinear programming, direct Pseudospectral (PS) methods have emerged as reliable direct methods for optimal control [8, 9, 10]. For quite some time, PS methods have been widely applied for complex scientific computation models involving differential equations, and PS methods have proven themselves as very efficient in solving these problems. However, only recently have PS methods intersected with nonlinear optimal control [11, 12]. This project uses a PS algorithm to numerically solve the problem of optimal control for the purpose of optimal UAV trajectory planning.

B. PROBLEM FORMULATION

This section will articulate how the PS method discretizes the problem of optimal control subjecting to nonlinear dynamics into a problem of finite dimensional nonlinear optimization, enabling the application of computational nonlinear programming to solve for the optimal control and optimal trajectories. Consider the following general optimal control problem:

$$\min[J(x(\bullet),u(\bullet))] = \int_{-1}^{+1} F(x(t),u(t))dt + E(x(-1),x(1))$$

subject to the following set of differential equations and initial conditions:

$$\dot{x} = f(x,u);\ \ x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

$$e(x(-1), x(1)) = 0;\ h(x(t), u(t)) \leq 0$$

where *x* is the state variable and *u* is the control input [7].

$$E(x(-1), x(1))$$

is the cost due to the endpoints,

$$e(x(-1), x(1)) = 0$$

is the condition at the endpoints, such as initial value, and

$$h(x(t), u(t)) \leq 0$$

is the set of state and control variable constraints. The PS method uses Sobolev spaces $W^{m,p}$, which involve functions $\xi(t)$, defined in $[-1,+1]$, whose *j*-th order weak derivative, $\xi^{(j)}$, lies in the space L^p for all $0 \leq j \leq m$ with the norm [7]

$$\left\|\xi\right\|_{W^{m,p}} = \sum_{j=0}^{m} \left\|\xi^{(j)}\right\|_{L^p} .$$

As previously mentioned, the PS method is an efficient direct method; this means that the actual optimal control problem, not the associated necessary conditions, is discretized to obtain a non-linear programming problem. Just as in any problem solved numerically, the accuracy of the PS method depends strongly on the method of approximation. Given a function $f(t):[a,b] \to \mathbb{R}$, one could conventionally approximate $f(t)$ by interpolating over uniformly spaced time nodes where $t_0 = a, t_1 = (b-a)/N, ..., t_N = b$, and N is equal to the number of interpolation points (or time nodes in the algorithm). However, it is widely known and proven that uniformly spaced points may produce numerical solutions with much higher approximation error, for the same number of points, than those calculated using other more sophisticated interpolation methods [7]. Furthermore, it is critical to emphasize that the number of interpolation points in calculating the solution to an optimal control problem is not only an issue of efficiency, but also of feasibility. A higher number of interpolation points results in a higher dimension in the non-linear programming model. If a particular model becomes too complex, it can easily overwhelm computational capabilities of an operating system; we obviously do not want this to happen and must carefully select an efficient interpolation method that can provide a low-error solution with relatively few nodes [7].

The PS model used for this project involves interpolating with Legendre-Gauss-Lobatto (LGL) quadrature nodes. These nodes, denoted by $\{t_0 = -1 < t_1 < t_2 < \ldots < t_{n-1} < t_n = 1\}$, are the roots of $\dot{L}_N$ where $\dot{L}_N$ is the N^{th}-order derivative of the Legendre Polynomial $L_N(t)$. Using this method, the range of integration is transformed universally to [-1,+1], which is the interval for Legendre Polynomials. Although the LGL interpolation points are not evenly spaced, they are symmetric about the midpoint 0 [7]. Figure 6 shows a range of LGL points for N = 16.

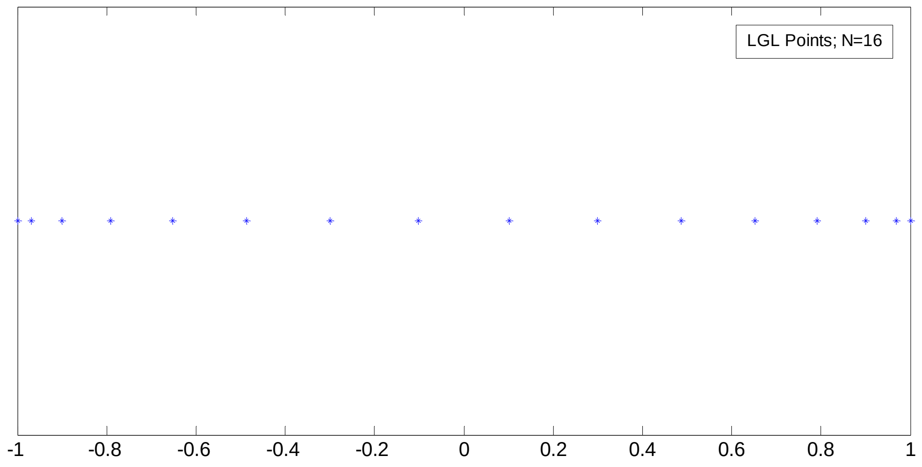

Figure 6. LGL Nodes, N = 16

PS methods are widely applied to numerically solve models using partial differential equations (PDEs). However, optimal control problems have several fundamental differences from the computation of PDEs. Solving optimal control problems asks for the collective and simultaneous solving of several different systems, including the differential equation governing the control system, the integration of the cost function, and the state-control constraints. Then these approximations are integrated together to form a problem of discrete nonlinear optimization which must be solved numerically to find the approximate optimal control [7].

In a PS optimal control method, the state and control functions, *x(t)* and *u(t)*, are approximated by N^th order Polynomials based on interpolation at selected LGL quadrature nodes. In the discretization, the state variables are approximated by the vectors

$$\overline{x}^{Nk} = \begin{pmatrix} \overline{x}_1^{\,Nk} \\ \overline{x}_2^{\,Nk} \\ \vdots \\ \overline{x}_r^{\,Nk} \end{pmatrix}$$

which is an approximation of $x(t_k)$. Also, the vector $\overline{u}^{Nk}$ is the approximation of $u(t_k)$. Consequently, a discrete estimation of the function $x_i(t)$ is the vector

$$\overline{x}_i^{\,N} = \left[\overline{x}_i^{\,N1},\ \overline{x}_i^{\,N2},\ \cdots,\overline{x}_i^{\,NN} \right].$$

A continuous estimate can be defined by a polynomial interpolation $x_i^N(t)$ where

$$x_i(t) \approx x_i^{\,N}(t) = \sum_{k=0}^{N} \overline{x}_i^{\,Nk} \Phi_k(t)$$

where $\Phi_k(t)$ is the Lagrange interpolating polynomial [13]. In contrast, the control input is approximated by the non-polynomial interpolation

$$u^N(t) = \frac{\dot{x}_r^{\,N}(t) - f(x^N(t))}{g(x^N(t))}.$$

In the notations, the discrete variables are denoted by letters with an upper bar, such as $\overline{x}_i^{Nk}$ and $\overline{u}^{Nk}$. If *k* in the superscript and/or *i* in the subscript are missing, then the notation represents a vector or matrix in which those particular indices run from their minimum to maximum values [7]. For example,

$$\overline{x}_i^{\,N} = \left[\overline{x}_i^{\,N1},\ \overline{x}_i^{\,N2},\ \cdots,\ \overline{x}_i^{\,NN} \right]$$

$$\overline{x}^{Nk} = \begin{pmatrix} \overline{x}_1^{\,Nk} \\ \overline{x}_2^{\,Nk} \\ \vdots \\ \overline{x}_r^{\,Nk} \end{pmatrix}$$

$$\overline{x}^{N} = \begin{pmatrix} \overline{x}_1^{\,N0} & \overline{x}_1^{\,N1} & \cdots & \overline{x}_1^{\,NN} \\ \overline{x}_2^{\,N0} & \overline{x}_2^{\,N1} & \cdots & \overline{x}_2^{\,NN} \\ \vdots & \vdots & \vdots & \vdots \\ \overline{x}_r^{\,N0} & \overline{x}_r^{\,N1} & \cdots & \overline{x}_r^{\,NN} \end{pmatrix}$$

$$\overline{u}^{N} = \left[\overline{u}^{N1},\ \overline{u}^{N2},\ \cdots,\ \overline{u}^{NN} \right]$$

where

$k = \{0,1,2,...,N\}$ is the interpolation point (node) number

$i = \{1,2,...,r\}$ is the state variable number

N = the total number of interpolation points (nodes)

r = the total number of state variables

and *k, i, N, and r* are the subscript and superscript definitions [7].

The analysis of spectral methods demonstrates that PS method is simple, accurate, and relatively fast in the estimation of smooth functions, integrations, and differentiations. These are all critical pieces for solving optimal control problems. The derivative of $x_i^N(t)$ at the LGL node t_k is easily computed by using matrix multiplication:

$$\left[\dot{x}_i^N(t_0)\,\dot{x}_i^N(t_1)\,\dot{x}_i^N(t_2)\cdots\dot{x}_i^N(t_N)\right]^T = D(\overline{x}_i^N)^T$$

where the $(N+1)\times(N+1)$ differentiation matrix *D* is defined by

$$D_{ik} = \begin{cases} \left(\dfrac{L_N(t_i)}{L_N(t_k)}\right)\left(\dfrac{1}{t_i - t_k}\right), if \;\; i \neq k \\ -\dfrac{N(N+1)}{4}, if \; i = k = 0 \\ \dfrac{N(N+1)}{4}, if \;\; i = k = N \\ 0, \qquad otherwise \end{cases}.$$

The cost functional $J\left[x(\cdot), u(\cdot)\right]$ is approximated by the Gauss-Lobatto integration rule,

$$J\left[x(\cdot), u(\cdot)\right] \approx \overline{J}^N\left(\overline{x}^N, \overline{u}^N\right) \;\; = \sum_{k=0}^{N} F\left(\overline{x}^{Nk}, \overline{u}^{Nk}\right) w_k + E\left(\overline{x}^{N0}, \overline{x}^{NN}\right)$$

where w_k are the LGL quadrature weights defined by

$$w_k = \frac{1}{\left(L_N(t_k)\right)^2}\left(\frac{2}{N(N+1)}\right).$$

The approximation is so accurate that it has no error if the integrand function is a polynomial of a degree less than *2N* [7].

In order to demonstrate the PS discretization, consider the following general nonlinear optimization problem:

Find $\overline{x}^{Nk} \in \mathbb{R}^r$ and $\overline{u}^{Nk} \in \mathbb{R}$, $k = \{0,1,\ldots,N\}$, that minimize

$$\overline{J}^N(\overline{x}^N, \overline{u}^N) = \sum_{k=0}^{N} F(\overline{x}^{Nk}, \overline{u}^{Nk}) w_k + E(\overline{x}^{N0}, \overline{x}^{NN})$$

subject to

$$\left\| \sum_{i=0}^{N} D_{ki} \overline{x}^{Ni} - f(\overline{x}^{Nk}, \overline{u}^{Nk}) \right\|_\infty \leq \delta$$

$$\left\| e(\overline{x}^{N0}, \overline{x}^{NN}) \right\|_\infty \leq \delta$$

$$h(\overline{x}^{Nk}, \overline{u}^{Nk}) \leq \delta$$

where δ is a small positive number as the relaxation of the constraints. Such relaxation is necessary as shown in [14]. It guarantees the feasibility of the problem.

$$\left\| \sum_{i=0}^{N} D_{ki} \overline{x}^{Ni} - f(\overline{x}^{Nk}, \overline{u}^{Nk}) \right\|_\infty \leq \delta$$

represents the discretization of the control system defined by the differential equation. Constraints are imposed and then tightened, as necessary, to define a search region within which nonlinear programming software packages can produce a feasible optimal control solution [7]. In [7], it is proven for feedback linearizable systems—a family of widely used control systems—that the value of the optimal cost of the discretized nonlinear optimization converges to the optimal cost of the original problem of optimal control. It is also proven that the method has a high order rate of convergence depending on the smoothness of the original problem. This result implies that the PS methods are convergent and numerically efficient.

In conclusion, this chapter has shown how PS optimal control methods allow a problem of optimal control subject to complex nonlinear dynamical and algebraic constraints to be discretized and solved numerically. The discretization works in a harmonic way for the multiple components in the problem, the cost function, the nonlinear control system, and the algebraic constraints. The efficient transition from continuous to discrete is crucial in the solution of the UAV optimal trajectory problem. Also, further analysis of References 7, 14 and 15 show that calculating optimal cost solutions using PS methods has a very high rate of convergence when compared to other methods; in fact, the Legendre PS method will have a faster convergence rate than any polynomial method [7, 14, 15].

IV. SIMULATIONS

A. INTRODUCTION

The objective for this thesis is to computationally find optimal trajectories and controls and obtain a unique solution for, at a minimum, the scenario discussed at the end of Chapter II: minimize flight time with no obstacles.

Before simulating this scenario, it is necessary to mesh two existing codes from two separate entities in order to use an optimizing algorithm in conjunction with the non-linear model of the X-Cell 60 Helicopter UAV. Prior to meshing these codes together, it is critical to ensure that they both function separately. Consequently, flight simulations are conducted using the non-linear helicopter and flight simulation codes provided by the team from the National University of Singapore, and a simple example scenario is optimized using an algorithm developed at the Naval Postgraduate School. Once the codes operate separately, the non-linear helicopter model is imported into the optimizing program to provide a single code that will optimize the trajectory for our selected Helicopter UAV model.

B. FLIGHT SIMULATION

The codes used from the National University of Singapore contain two key files: 1) The nonlinear helicopter model and 2) The flight simulation code (which takes the non-linear model and inputs it into a fourth order Runge-Kutta Method loop to fly over a set number of time steps). Initially, the code designers set the control variables δ_{lat}, δ_{lon}, and δ_{col} as fixed values throughout all time steps and the δ_{ped} control variable to change linearly with respect to the Intermediate State in Yaw Rate Gyro Dynamics state variable. These control variable settings ensured flight stability and are set in a manner to fly the UAV on a relatively straight path in the *y* direction (due East). Figures 7 and 8 show the positions and velocities, respectively, in the *x, y,* and *z* directions over time with the parameter set by the code designers.

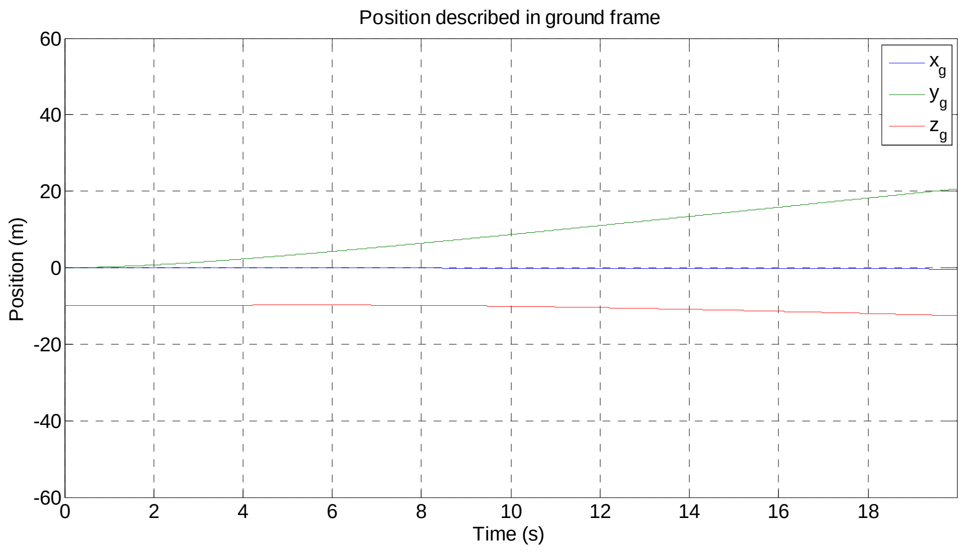

Figure 7. Flight Simulation, positions (x,y,z) with respect to time

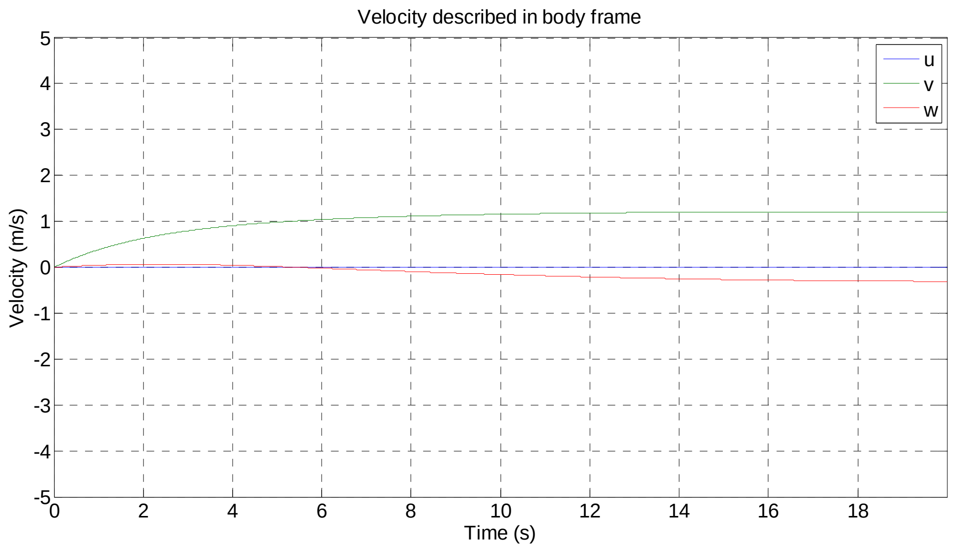

Figure 8. Flight Simulation, velocities (u,v,w) with respect to time

As shown in Figures 7 and 8, the Flight Simulation code, using the non-linear model, produces a stable flight for the Helicopter UAV; therefore, the non-linear model should behave properly when integrated into the optimization algorithm.

C. OPTIMIZATION EXAMPLE

In order to gain a better understanding for the optimization program, it was essential to run an elementary example prior to solving such a complex system as our Helicopter UAV model. The initial example is a simple system with two state variables and one control variable. The example problem is

$$\begin{cases} \dot{x} = y \\ \dot{y} = y + u \end{cases}$$

where u is the control variable and the performance measure is

$$\min[J] = \int_0^1 (x^2 + y^2 + .005u)\, dt$$

subject to the constraint

$$y \leq 8\left(t - \frac{1}{2}\right)^2 - \frac{1}{2}$$

and the initial conditions

$$\begin{cases} x(0) = 0 \\ y(0) = -1 \end{cases}$$

where $t \in [0,1]$.

It turns out that the optimization algorithm provides an optimal solution to this problem. This is determined by the Exit Codes; these codes are built into the program to show that a solution is optimal or to help refine possible reasons for non-optimal solutions. In this particular case, the Exit Code is 1—an optimal solution with all conditions satisfied [16]. Figure 9 depicts the control variable values throughout the time interval, and Figure 10 shows the *x*, *y*, and *constraint* values throughout the time interval. It is clear from Figure 10 that the state variable values follow the constraint. All of the outputs from the optimization algorithm determine that the value $J = .1803$ is a minimum value.

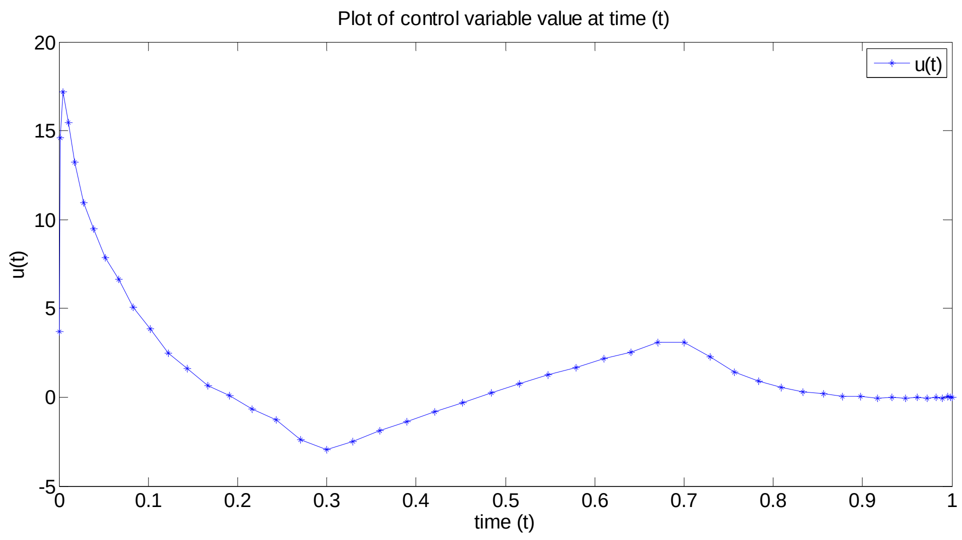

Figure 9. Control Variable (u) values at (t) for the Optimization Example

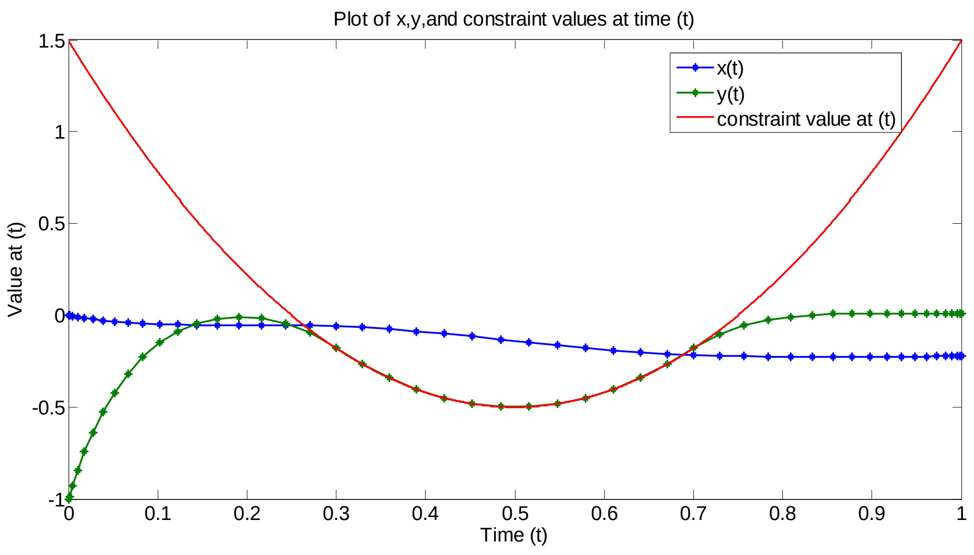

Figure 10. x, y, and constraint values at (t) for the Optimization Example

D. OPTIMAL CONTROL OF THE HELICOPTER UAV

Since both codes have separately provided successful results, it is time to mesh the codes and begin working through some examples using our non-linear Helicopter UAV model. It is crucial for us to understand that simulations are testing the optimal control algorithm and the non-linear UAV model. The UAV model has been verified for stability and simulation of smooth curve flight but not necessarily for use in optimal control. Any simulation errors or inconsistencies could result from incompatibilities in either code.

For all optimization simulations, 30 LGL time nodes are used. This number of nodes seem to provide the best results throughout simulations. The control variables, as given in the non-linear helicopter model, have non-dimensional values and are bounded in the following manner:

$$\delta_{lat} \in [-1, 1]$$

$$\delta_{lon} \in [-1, 1]$$

$$\delta_{col} \in [-1, 0]$$

$$\delta_{ped} \in [-1, 1].$$

1. Minimize Time, No Obstacles

For the time minimization performance measure, four simple examples are run:

Example 1: The UAV travels from the point (0, 0, -10) to the point (10, 0, -10) (the initial and final position values in the *y* and *z* directions, respectively, are the same)

Example 2: The UAV travels from the point (0, 0, -10) to the point (10, 10, -10) (the initial and final position values are the same in only the *z* directions)

Example 3: The UAV travels from the point (0, 0, -10) to the point (50, 0, -10) (same conditions in the *y* and *z* directions with the distance travelled in the *x* direction increased from 10 to 50 units)

Example 4: The UAV travels from the point (0, 0, -10) to the point (50, 50, -10) (the total distance travelled in both the *x* and *y* directions is increased from 10 to 50 units).

All four examples have the same initial conditions for the state variables (all values are zero, except for p_z, which is initially -10; this implies that the helicopter begins at a position 10 units above the ground). Since the objective is to minimize time, the integral for the performance measure is simply

$$\min[J] = \int_0^{t_f} 1 dt$$

where the solution simplifies to t_f.

For Example 1, 12 of the state variables have fixed boundary conditions at t_f; they are annotated in the vector equation

$$\begin{pmatrix} p_{x_f} \\ p_{y_f} \\ p_{z_f} \\ u_f \\ v_f \\ w_f \\ \Phi_f \\ \theta_f \\ \Psi_f \\ q_f \\ r_f \\ s_f \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Since there are no obstacles impeding the trajectory, the expectation is that the solution should be a relatively straight flight path in the *x*-direction with a smooth acceleration for the first half of the flight followed by a smooth deceleration for the second half in order to return to a *u* and *v* value of 0 at t_f. As shown in Figures 11 and 12, the actual solution provided by the algorithm adheres closely to our predictions.

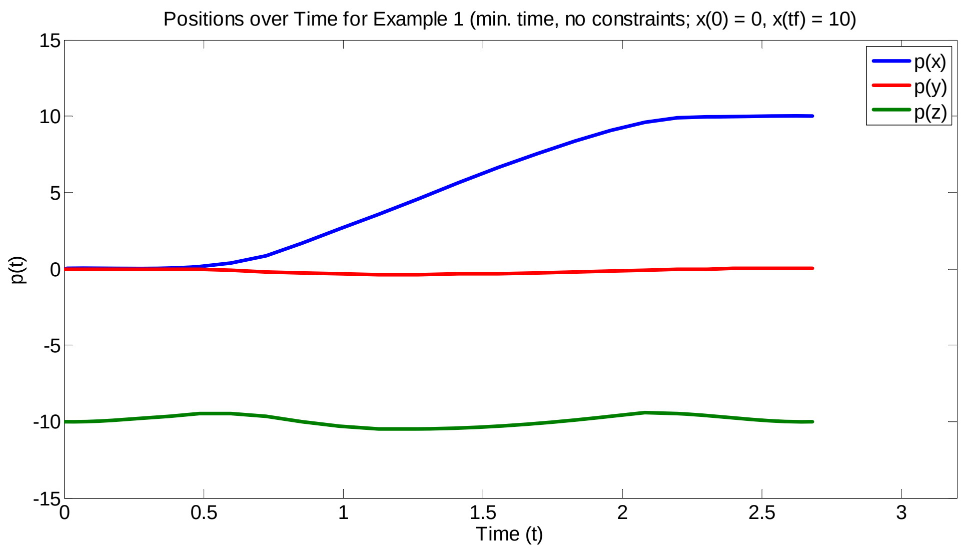

Figure 11. Positions over time for Example 1

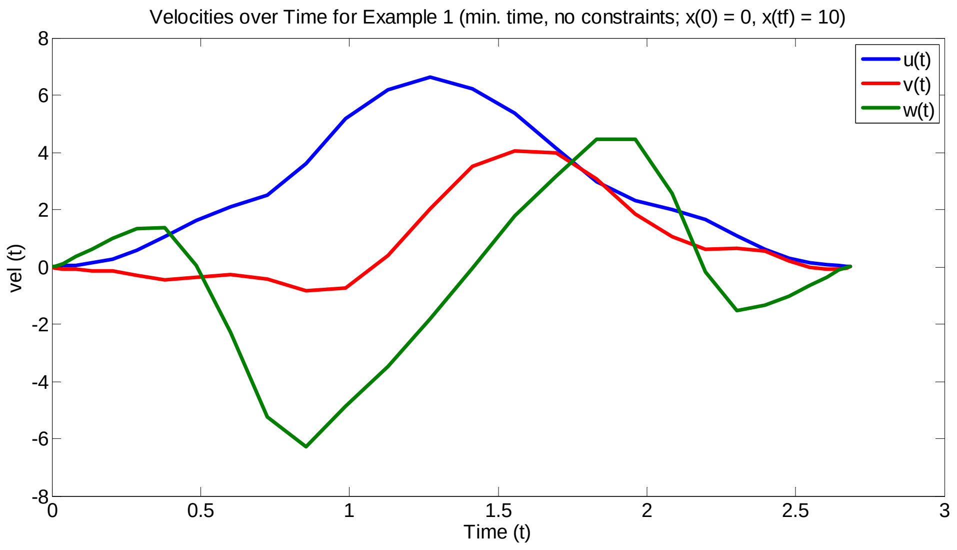

Figure 12. Velocities over time for Example 1

The minimum time required is

$$\min[J] = t_f = 2.6905 \,.$$

Figure 13 depicts the four control variable values throughout the flight time span. In the control variable plots for all examples in this chapter, the following substitutions apply:

$$U1 = \delta_{lat}$$

$$U2 = \delta_{lon}$$

$$U3 = \delta_{col}$$

$$U4 = \delta_{ped} \,.$$

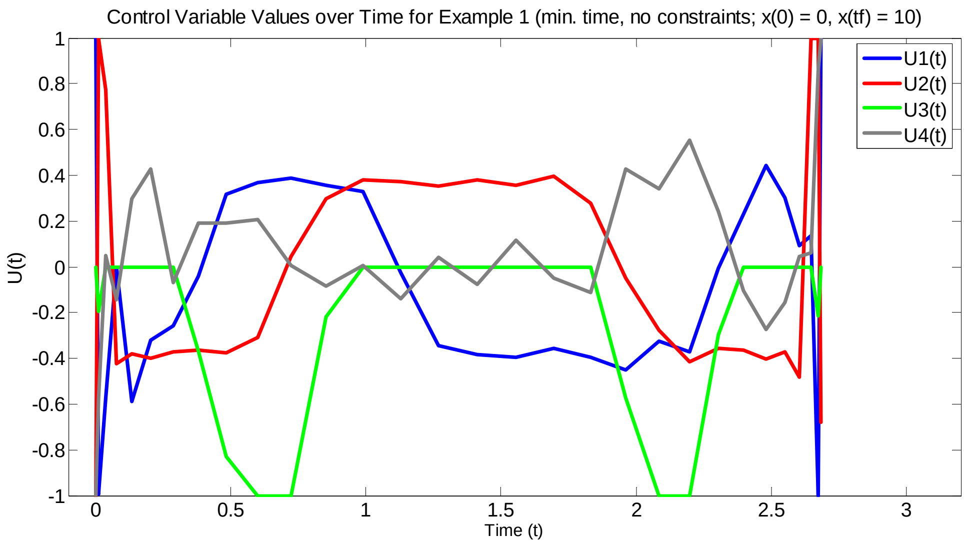

Figure 13. Control variable values over time for Example 1

The solution value and the position plots in Figure 11 seem to hint that our solution is valid and quite possibly a true optimal solution to Example 1. However, the velocity and control variable plots show some unexpected inconsistencies that contradict the Exit Code from the program output (which suggests that the solution is truly optimal). The Exit Code, 1, as defined in the algorithm's user guide, tells us that the simulation has produced an optimal value for the performance measure [16]. Figure 12 shows an unexpected range in velocities in the z direction—there should be very little velocity in the y or z direction to fly from 0 to 10 in only the x direction. Also, Figure 13 demonstrates high levels of instability in the control variable values near t_0 and t_f. These observations from the simulation results lead us to believe that the trajectory does not obey the dynamical rules of the system. This issue will be discussed later in this section.

For Example 2, 12 of the state variables have fixed boundary conditions at t_f; they are annotated in the vector equation

$$\begin{pmatrix} p_{x_f} \\ p_{y_f} \\ p_{z_f} \\ u_f \\ v_f \\ w_f \\ \Phi_f \\ \theta_f \\ \Psi_f \\ q_f \\ r_f \\ s_f \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Since, again, there are no obstacles involved, the expectation is that the solution should be a relatively straight flight path in the *xy*-direction with a smooth acceleration for the first half of the flight followed by a smooth deceleration for the second half in order to return to a *u* and *v* value of 0 at t_f. One would expect a similar position and velocity curve in the *x* and y directions. Figures 14 and 15 show that the actual solution provided by the algorithm adheres closely to our predictions.

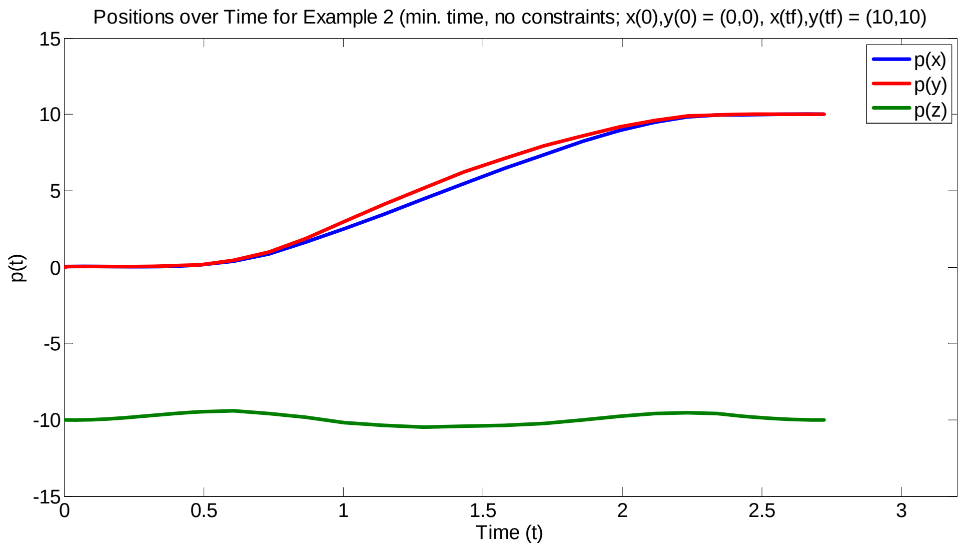

Figure 14. Positions over time for Example 2

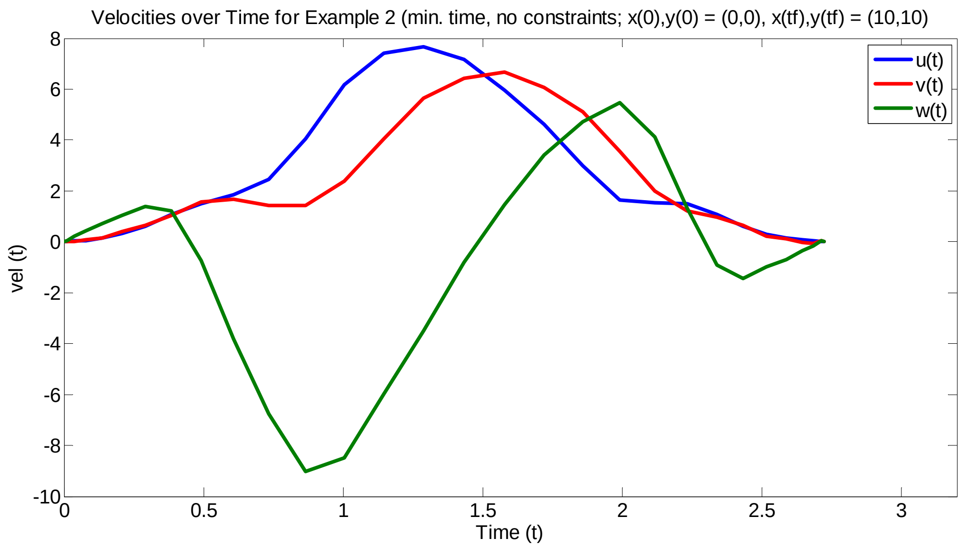

Figure 15. Velocities over time for Example 2

The minimum time required is

$$\min[J] = t_f = 2.7255 .$$

Figure 16 depicts the four control variable values throughout the flight time span.

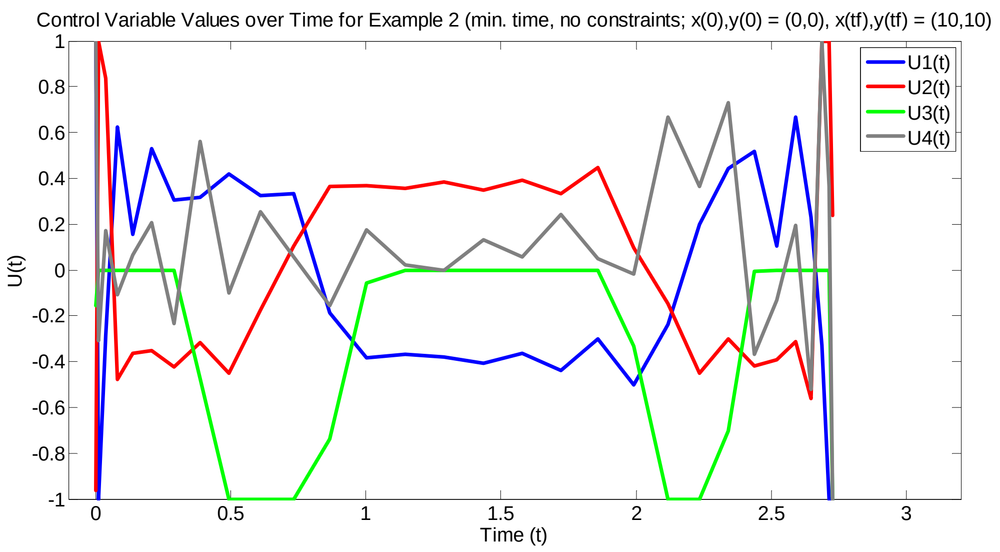

Figure 16. Control variable values over time for Example 2

The solution value, the position plot in Figure 14, and the velocity plot in Figure 15 (in the *x* and y directions) all seem to hint that our solution is valid and quite possibly a true optimal solution to Example 2. However, the velocity in the *z* direction and control variable plots show some unexpected inconsistencies that help to explain the Exit Code from the program output (which suggests that the solution is not optimal). Example 2 had a different Exit Code output from Example 1 (Exit Code 41) which, according to the user's guide, means that our solution is nearly optimal but encounters numerical difficulties when the sub-optimal solution cannot be improved [16]. This is apparent from the simulation run times: 10 to 15 seconds for Exit Code 1 but 400 to 1300 seconds for Exit Code 41. Similar to Example 1, Figure 15 shows an unexpected range in velocities in the *z* direction—there should be very little velocity in the *z* direction to fly from 0 to 10 in the *x* and *y* directions—and Figure 16 demonstrates high levels of instability in the control variable values near t_0 and t_f which could directly influence the optimality of the solution.

For Example 3, 12 of the state variables have fixed boundary conditions at t_f; they are annotated in the vector equation

$$\begin{pmatrix} p_{x_f} \\ p_{y_f} \\ p_{z_f} \\ u_f \\ v_f \\ w_f \\ \Phi_f \\ \theta_f \\ \Psi_f \\ q_f \\ r_f \\ s_f \end{pmatrix} = \begin{pmatrix} 50 \\ 0 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Similar to Example 1, the UAV should fly a relatively straight path in the *x*-direction with a smooth acceleration for the first half of the flight followed by a smooth deceleration for the second half in order to return to a *u* and *v* value of 0 at t_f. As shown in Figure 17, the actual solution provided by the algorithm adheres closely to our predictions. However, Figure 18 shows unexpected changes in velocity in both the *y* and *z* directions.

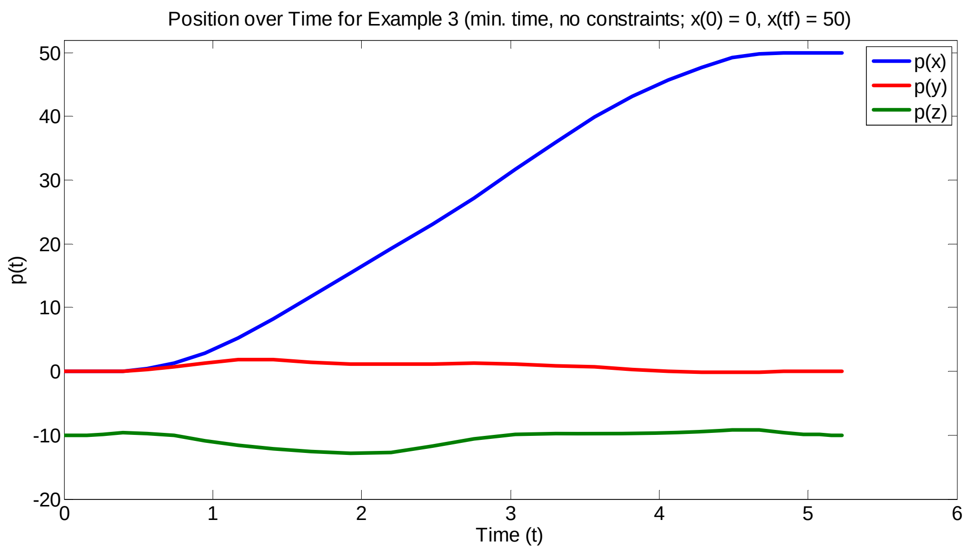

Figure 17. Positions over time for Example 3

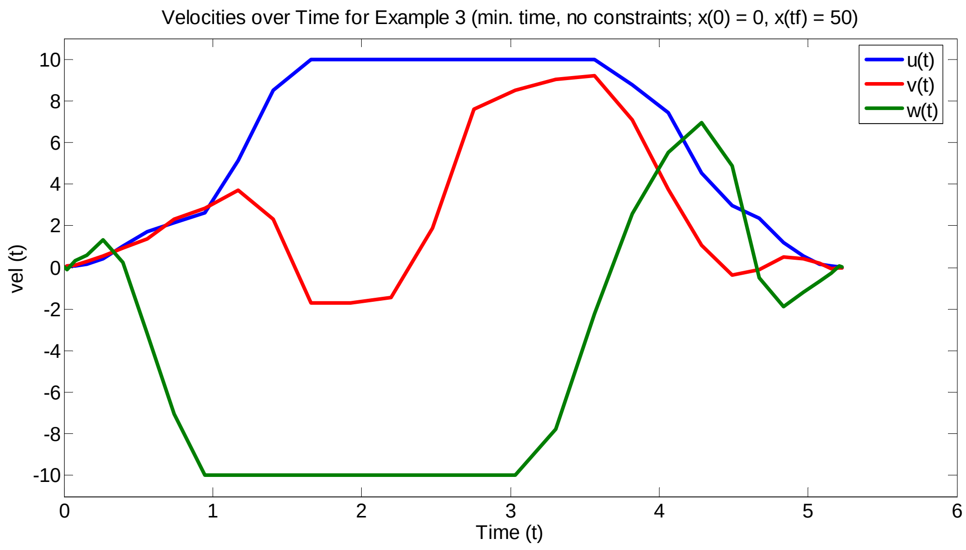

Figure 18. Velocities over time for Example 3

Figures 17 and 18 clearly do not match up for their *y* and *z* components. Based on Figure 18, there should be substantial position changes in all three directions in Figure 17. This is not the case; however, the simulation achieves an Exit Code 1 value that implies the following is an optimal solution [16]:

$$\min[J] = t_f = 5.2311.$$

However, the plot discrepancies suggest an issue with the non-linear UAV model portion of the code. Figure 19 demonstrates similar endpoint instability for the control variable as were seen in Examples 1 and 2; however, the instability is not nearly as drastic as in the first two examples.

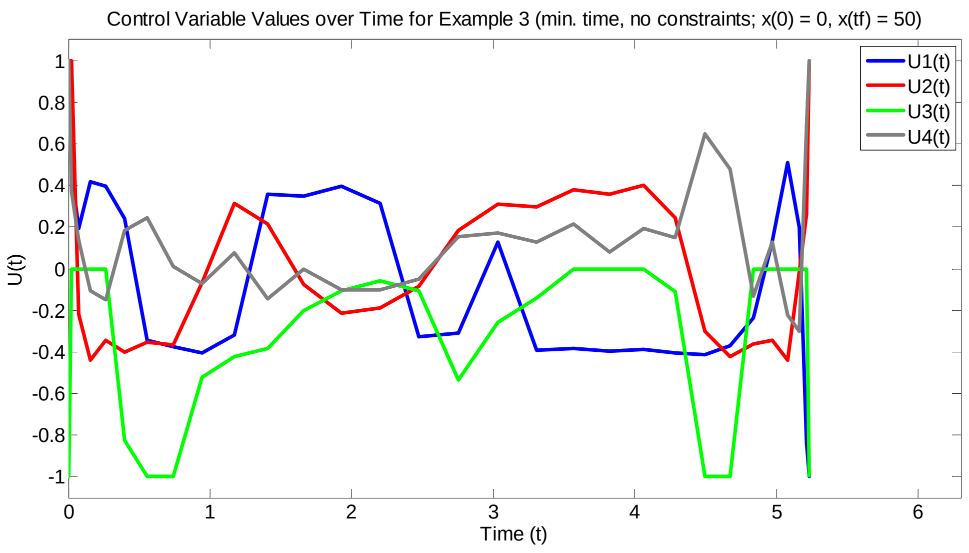

Figure 19. Control variable values over time for Example 3

For Example 4,12 of the state variables have fixed boundary conditions at t_f; they are annotated in the vector equation

$$\begin{pmatrix} p_{x_f} \\ p_{y_f} \\ p_{z_f} \\ u_f \\ v_f \\ w_f \\ \Phi_f \\ \theta_f \\ \Psi_f \\ q_f \\ r_f \\ s_f \end{pmatrix} = \begin{pmatrix} 50 \\ 50 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Since, again, there are no obstacles involved, the expectation is that the solution should be a relatively straight flight path in the *xy*-direction with a smooth acceleration for the first half of the flight followed by a smooth deceleration for the second half in order to return to a *u* and *v* value of 0 at t_f. Just as in Example 2, one would expect a similar position and velocity curve in the *x* and y directions. Figures 20 and 21 show that the actual solution provided by the algorithm adheres closely to our predictions.

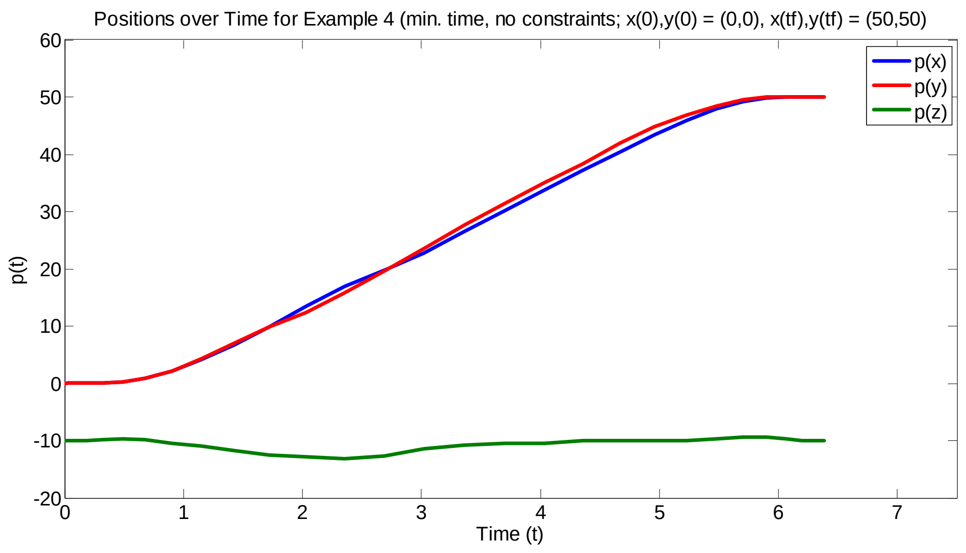

Figure 20. Positions over time for Example 4

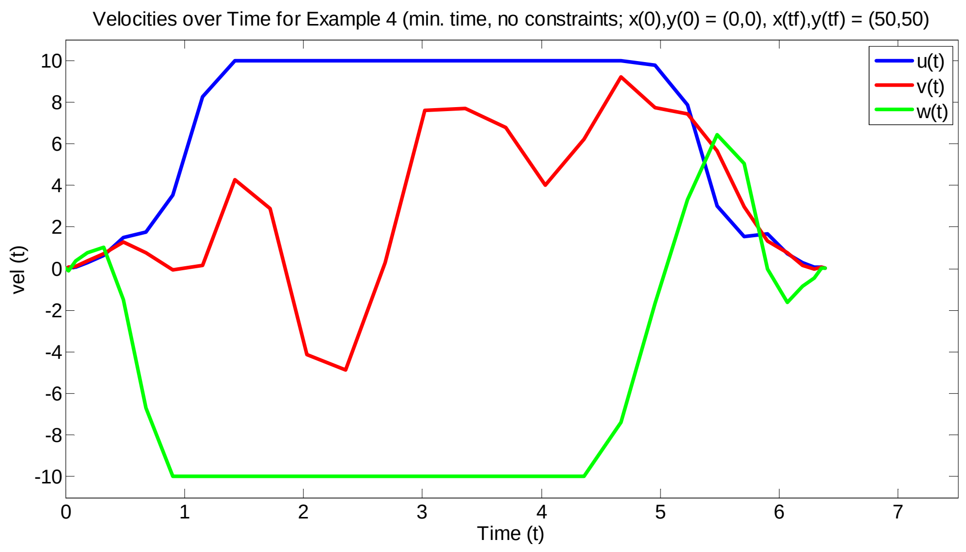

Figure 21. Velocities over time for Example 4

The minimum time required is

$$\min[J] = t_f = 6.3853.$$

Figure 22 depicts the four control variable values throughout the flight time span.

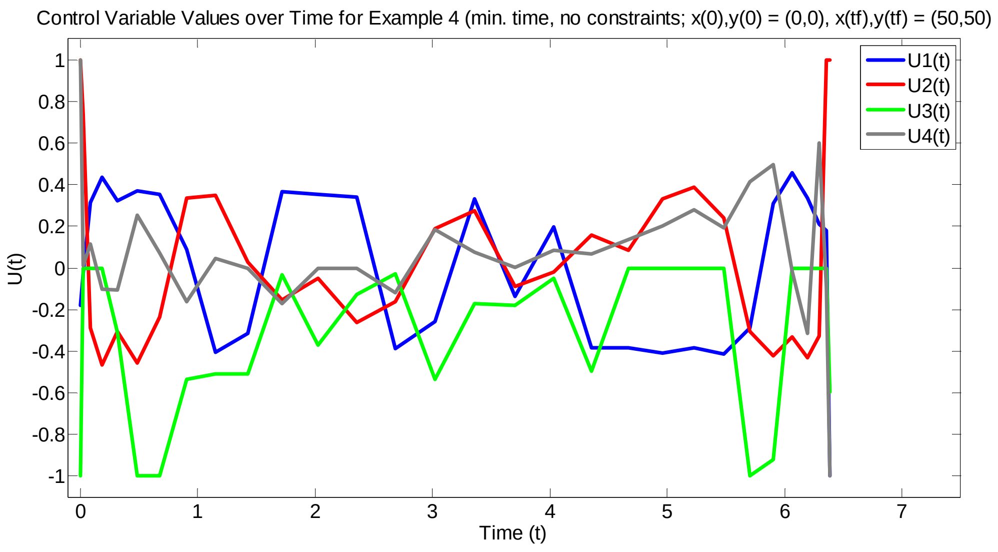

Figure 22. Control variable values over time for Example 4

Just as in Example 2, the solution value, the position plot in Figure 20, and the velocity plot in Figure 21 (in the *x* direction) all seem to hint that our solution is valid and quite possibly a true optimal solution to Example 4. However, the velocity in the *y* and *z* directions and control variable plots show some unexpected inconsistencies that help to explain the Exit Code of 41 from the program output (which, just as in Example 2, suggests that the solution is not optimal) [16]. The inconsistencies between the position and velocity plots are very similar to those in Example 3. The control variable values in Figure 22 show that the endpoint instability is not nearly as drastic as in Examples 1 and 2 but similar to that shown in Example 3. It seems that lengthening that path tends to smooth the control variable variation to some degree.

The previous four simple examples have demonstrated that the meshing of the nonlinear UAV model and the optimization algorithm codes is functional and effectively simulates a three dimensional trajectory while outputting an optimal or near optimal solution for minimal flight time for several different sets of initial and final conditions. However, there are several issues that must be worked out in the future. The inconsistency between the path and velocity trajectories implies that discrepancies exist when the UAV model is integrated with the optimal control software package. It is a known fact that computational optimal control requires higher accuracy than what is needed in the integration of ODEs. As recently as two weeks ago, the UAV Laboratory of the National University of Singapore developed a new model of the same UAV system. According to their analysis and simulations, the accuracy, when comparing to experimentation data, is much higher then the old model that is being used in our program. We are advised to use the new model in all simulations. The next step of research is to integrate the new model into the package developed in this thesis to achieve improved simulation results. In addition, various numbers of nodes and scenarios are to be numerically tested and analyzed.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE RESEARCH

A. CONCLUSIONS

This thesis project has been a very interesting look into the value of numerical methods for solving complex problems. There are several findings, some positive and some negative, that have impacted the results and, consequently, have set the ground work for a plethora of future research.

First, and foremost, the meshing of two models from two different software packages into one set of optimization codes was effective and produced optimal results for several simple examples. The organization of the optimization codes into five separate user input file—one for the performance measure, one to manage the system of dynamic and control equations, one to manage the constraints, one to manage the initial and final conditions of the equation system, and one main code to run the algorithm—proved very useful in segregating and resolving errors during the early coding phases. Also, this organization of the codes made it quite simple to set up new examples for analysis.

Interestingly, the optimization code is very sensitive to bounds placed on the variables. It was an obvious assumption that bounds too restrictive in nature would make the problem infeasible. As a result, the program cannot calculate an optimal solution. However, if the range between the upper and lower bound is too large, it may take a long time for the program to converge. In worse cases, the program may diverge. This sensitivity forces the user to carefully select the bounds and continue refining them, in some cases, to allow the program to run successfully.

Since some examples resulted in an optimal solution and some did not, it seems logical that there are some issues with both models that need to be worked out in order to make the optimization algorithm and nonlinear model operate in harmony. Since the non-linear model is not originally designed for optimization, this is a sensible conclusion. Some of the constant values that create an ideal environment for flight simulation may not necessarily do so for optimization of a flight trajectory. The fact that it works in certain cases is a sign that the nonlinear model should not have to be altered much to operate much more effectively within the optimization algorithm for additional simulations.

Another disparity that hints toward non-conformity within the nonlinear UAV model is the issue of disagreement between one or more of the velocity components over time as compared to their respective position components. The fact that, in some cases, the optimization algorithm outputs an Exit Code 1 (an optimal solution output) while the plots do not make sense, hints that there is a discrepancy when the UAV model is integrated with the optimal control program. These inconsistencies must be adjusted before one can simulate and optimize more complex trajectories.

The final finding, which would make the examples in this thesis very difficult to transition to real-life flight simulations, is the erratic nature of the control variable values, especially near the initial and final time steps. It seems that the actual helicopter UAV would fly in a very unstable fashion with the plotted control variable inputs. Smoothing these input values would make the model a more realistic feat for actual test flight trajectory analysis.

Overall, this project was an interesting and enjoyable research experience where I was able to learn a great deal about a subject of which I previously knew almost nothing about. The meshing of two codes into one functional optimization algorithm with the ability to simulate an extremely complex flight trajectory and output an optimal solution was a great success, and I look forward to pursuing future endeavors on this project with my advisor.

B. FUTURE RESEARCH

There is a tremendous amount work that can be done to further the research in this thesis, and I plan on pursuing some of this work with Dr. Kang in the future. Some future endeavors include the following:

- Working with the recently improved UAV model to get it to conform better to the optimization algorithm
- Applying the optimization algorithm with this particular helicopter UAV to more complex flight trajectories such as collision avoidance in obstacle dense environment
- Optimizing other performance measures
- Applying the same optimization algorithm to other UAV types and actual helicopters.

The modeling of these scenarios could prove invaluable to the United States Military's ability to maximize the effectiveness of limited UAV resources in combat zones and could eventually model trajectories minimizing the risk to pilots and aircraft for helicopters of different size and purpose.

Improving both the UAV model and the optimization algorithm will enable the program to calculate solutions for more complex trajectory simulations. For example, a user could add obstacles to the UAV's path, which, up to this point, we have not been able to code effectively. Also, one could track a path (potentially a hostile person or vehicle on the ground) with minimum variance from that path. This model would be incredibly useful in urban environments. In addition to minimizing, a user could opt to try to maximize a UAV's capability, such as the number targets or amount of ground area observed within a given flight time window. In the examples for this thesis, we minimized the flight time, but additional parameters could be added to the model to minimize fuel consumption or flight cost. These are just a few examples of more complex trajectories with alternate performance measures for realistic UAV operations.

Trajectory optimization for helicopter UAVs definitely has real-world applications, which means many more ways to continue building on this project. Hopefully, these applications will provide the military with plausible options for maximizing effectiveness and minimizing risk and cost on the future battlefield.

LIST OF REFERENCES

[1] M. Shanmugavel, A. Tsourdos, R. Zbikowski, and B. A. White, “3D Dubins Sets Based Coordinated Path Planning for Swarms of UAVs,” *American Institute of Aeronautics and Astronautics*, August 2006.

[2] U. Zengin, and A. Dogan, “Probabilistic Trajectory Planning for UAVs in Dynamic Environments,” *American Institute of Aeronautics and Astronautics*, September 2004.

[3] M. Hurni, “An Information-centric Approach to Autonomous Trajectory Planning Utilizing Optimal Control Techniques,” *Dissertation, Department of Mechanical Engineering, Naval Postgraduate School*, September 2009.

[4] J. Yang, Z. Qu, J. Wang, K. L. Conrad, and R. A. Hull, “Real-time Obstacles Avoidance for Vehicles in the Urban Grand Challenge,” *Journal of Aerospace Computing, Information, and Communication*, vol. 4, December 2007.

[5] G. Cai, B. M. Chen, T. H. Lee, and K. Y. Lum, “Comprehensive Nonlinear Modeling of an Unmanned Aerial Vehicle Helicopter,” *Journal of the American Helicopter Society*, December 2009.

[6] E. R. Pinch, *Optimal Control and the Calculus of Variations*, *Oxford University Press*, 1993.

[7] W. Kang, “Rate of Convergence for a Legendre Pseudospectral Optimal Control of Feedback Linearizable Systems,” *Journal of Control Theory Applications*, vol. 7, no. 1, 2009, pp. 123–137.

[8] J. T. Betts, “Practical Methods for Optimal Control Using Nonlinear Programming,” *SIAM,* 2001.

[9] J. T. Betts, “Survey of Numerical Methods for Trajectory Optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, 1998, pp. 193–207.

[10] E. Polak, “Optimization: Algorithms and Consistent Approximations,” *Springer-Verlag*, Heidelberg, 1997.

[11] G. Elnagar and M. A. Kazemi, “Pseudospectral Chebyshev Optimal Control of Constrained Nonlinear Dynamical Systems,” *Computational Optimization and Applications*, 1998, pp. 195–217.

[12] F. Fahroo and I. M. Ross, “Costate Estimation by a Legendre Pseudospectral Method,” *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Boston, MA, 10–12 August 1998.

[13] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, “Spectral Method in Fluid Dynamics,” *New York: Springer–Verlag*, 1998.

[14] Q. Gong, W. Kang, and I. M. Ross, “A Pseudospectral Method for the Optimal Control of Constrained Feedback Linearizable Systems,” *IEEE Trans. Automat. Contr.*, vol. 51, no. 7, 2006, pp. 1115–1129.

[15] W. Kang, Q. Gong, and I. M Ross, “On the Convergence of Nonlinear Optimal Control Using Pseudospectral Methods for Feedback Linearizable Systems,” *International Journal of Robust and Nonlinear Control*, vol. 17, 2007, pp. 1251–1277.

[16] P. E. Gill, W. Murray, and M. A. Saunders, “User’s Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming,” February 2006.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Wei Kang
   Department of Applied Mathematics
   Naval Postgraduate School
   Monterey, California

4. Professor Hong Zhou
   Department of Applied Mathematics
   Naval Postgraduate School
   Monterey, California

5. Professor Carlos Borges
   Department of Applied Mathematics
   Naval Postgraduate School
   Monterey, California

6. COL Michael Phillips
   Department of Mathematical Sciences
   United States Military Academy
   West Point, New York

7. MAJ Benjamin Gatzke
   Department of Mathematical Sciences
   United States Military Academy
   West Point, New York